NTRCI

NATIONAL TRANSPORTATION RESEARCH CENTER, INCORPORATED
University Transportation Center

U18: Traffic Signal Safety (Phase B)

This project was funded by the NTRCI University Transportation Center under a grant from the U.S. Department of Transportation Research and Innovative Technology Administration (#DTRT06G-0043)

Itamar Arel and Thomas Urbanik II
The University of Tennessee

August 2009

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle **U18: Traffic Signal Safety (Phase B)**		5. Report Date **August 2009**
		6. Performing Organization Code
7. Author(s) **Itamar Arel and Thomas Urbanik II – The University of Tennessee**		8. Performing Organization Report No.
9. Performing Organization Name and Address **National Transportation Research Center, Inc.** **University Transportation Center** **2360 Cherahala Blvd.** **Knoxville, TN 37932**		10. Work Unit No. (TRAIS) 11. Contract or Grant No. **DTRT06G-0043**
12. Sponsoring Agency Name and Address **U.S. Department of Transportation** **Research and Innovative Technology Administration** **1200 New Jersey Avenue, SE** **Washington, DC 20590**		13. Type of Report and Period Covered **Final Report September 2008 - August 2009** 14. Sponsoring Agency Code **RITA**
15. Supplementary Notes		

16. Abstract

Efficiently scheduling traffic, particularly heavy vehicles, remains a key challenge in transportation engineering. This project has focused on the development of a novel traffic–signal-control methodology to improve the safety of heavy vehicles on high-speed approaches to signalized intersections. The approach makes use of wireless communications between the heavy vehicle and the traffic-signal controller. The project builds upon the Trusted Truck® infrastructure in order to have a more cost-effective deployment. The project also considered existing intelligent transportation system standards (e.g. National Transportation Communications for ITS Protocol [NTCIP]) as a means of establishing a practical implementation path.

17. Key Word	18. Distribution Statement
Traffic Signal Safety, heavy vehicle, wireless communications, Trusted Truck®, traffic-signal infrastructure, signal-scheduling algorithm	**No restrictions**

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	**Unclassified**	**45**	

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Table of Contents

List of Figures

Executive Summary

The project successfully demonstrated a novel traffic-scheduling algorithm, implemented using NTCIP, making the concept portable to multiple vendors. The scheme is adaptable with future flexibility in place as more experience is obtained, and does not require any change to existing traffic-control infrastructure. In cases of high traffic volume, the LQF-MWM algorithm yields an average vehicle delay that is half of that observed when a traditional National Electrical Manufactures Association (NEMA) controller operates the signals using standard NEMA sequencing. In summary, this two-year effort successfully achieved its objective in assessing the feasibility of collaborative heavy vehicle/traffic signal operation using real control-logic hardware. It is concluded that improved heavy-traffic-flow efficiency and safety is attainable with the proposed framework.

Chapter 1 – Background

Efficiently scheduling traffic, particularly heavy vehicles, remains a key challenge in transportation engineering. This project has focused on the development of a novel traffic–signal-control methodology to improve the safety of heavy vehicles on high-speed approaches to signalized intersections. The approach makes use of wireless communications between the heavy vehicle and the traffic-signal controller. The project builds upon the Trusted Truck® infrastructure in order to have a more cost-effective deployment. The project also considered existing intelligent transportation system standards (e.g. National Transportation Communications for ITS Protocol [NTCIP]) as a means of establishing a practical implementation path.

Because the existing traffic-signal infrastructure does not typically have the information or logic necessary to implement truly intelligent decisions, it is necessary to re-engineer a solution that provides the functionality to achieve the objective.

During the first year of the project, a signal-scheduling algorithm was developed to maximize truck throughput while minimizing the delay experienced by vehicles across the intersection. In particular, truck safety is a key consideration. This is achieved by allowing the algorithm to consider trucks that are both waiting for service as well as approaching the intersection. Safety is increased when trucks are stopped less frequently at high-speed intersections. By considering the requests (i.e. vehicles awaiting service) on all approaches of the intersection and by issuing a green light to the most-urgent combination of flows, the algorithm achieves its high performance. As a result of applying a more intelligent traffic-scheduling controller, we observed lower average vehicle delay as well as reduced truck stops, across different traffic scenarios. The second year addressed the hardware-in-the-loop implementation of the proposed framework, including more-elaborate simulation settings and development of related communication protocols. The following sections describe the work completed during the two years of the project.

Chapter 2 – System Model

Intersection Configuration

The intersection under consideration is illustrated in Figure 1. This is a four-approach intersection with through lanes (that also serve right turns) and exclusive left-turn lanes. Each phase in this intersection is labeled following the NEMA (National Electrical Manufacturers Association) convention. This intersection is an adequate test case, not only because it appears often in real-world traffic networks, but also because its symmetry allows for a fairly straightforward analysis. It should be noted, however, that this control technique may be applied to any intersection layout.

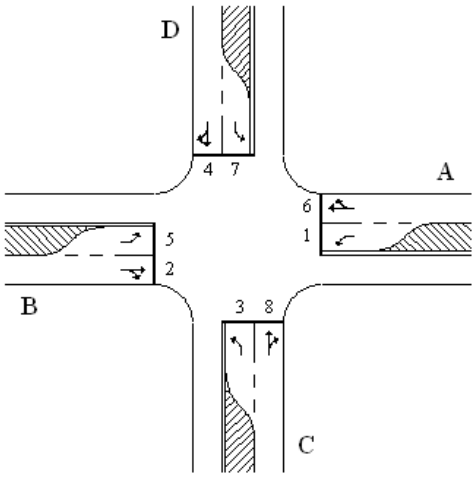

Figure 1. Diagram. An isolated intersection.

All approaches to the study intersection are 1000 meters long. The long approaches help to ensure that arriving traffic is distributed properly, and that delayed vehicles exceed the approach length. This is particularly important for simulation runs with traffic flow approaching the saturation level. The left-turn lanes provide 100 meters of vehicle queue space to help prevent blocking from occurring at the lane branching point. The distribution of vehicle route selection (i.e. left, straight, or right) is identical for all approaches. These parameters have been chosen arbitrarily, but the goal is to simulate a somewhat realistic case simply to demonstrate the operation of the control method. It is important to remember that this method will work for any intersection under any load conditions, so long as certain criteria are met (as discussed in the section on the Scheduling Algorithm below).

Performance Measures

Fundamental measures for evaluating the performance of a traffic controller (particularly at an isolated intersection) include: vehicle delay, traffic throughput, vehicle stops, and average queue

size. Analyzing the overall delay experienced by a vehicle that has traversed the network is a direct indication of how long the vehicle has had to wait at the intersection prior to traversing it. Throughput measures the number of vehicles per hour that pass through an intersection, which is also indicative of the overall controller performance. Vehicle stops refers to the number of times that a vehicle must come to a stop while attempting to traverse a traffic network. Naturally, decreasing the number of stops needed is a primary goal of traffic controllers. However, the queue size is the most important measure studied in this work. As expressed by Little's theorem [1], the average delay experienced by the vehicles in the network is directly proportional to the average queue size [2]. Thus, minimizing the average queue sizes can, in general, minimize the average vehicle delays.

Although minimizing the queue sizes is a primary motivation of the Longest Queue First Maximal Weight Matching (LQF-MWM) algorithm, the overall vehicle delay is the measure that we use to compare the performance of the control methods used. Given the inherent symmetry of the network under consideration and the identical speed of the vehicles, if there are no queuing delays, then all of the vehicles will have an identical time delay when traversing the intersection, regardless of the path taken. Therefore, the queuing time delay is obtained by subtracting the best-case time consumed traversing the network (i.e. no intersection delay) from the overall time that each individual vehicle has spent in the network.

Traffic Cycle Attributes

In order for the intersection to operate properly, and in an effort to ensure the safety of the vehicles, the traffic cycle length must be valid. Commonly, the cycle consists of phases placed in a particular order—with each interval given some amount of cycle time. The ring diagram for the test intersection is depicted in Figure 2. In this diagram, time progresses from left to right, indicating that the left-turn phases become active before the corresponding straight/right phases for each approach (referencing the phase numbering of Figure 1).

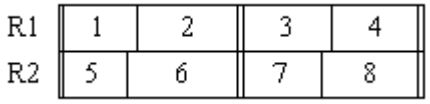

Figure 2. Diagram. The eight-phase ring diagram for the intersection considered.

Phases are said to be compatible if they can be green concurrently without creating traffic-flow conflicts. A vertical barrier separates the East-West phases from the North-South phases. Each of the phases is compatible with the phases above or below itself and on the same side of the barrier. All other phase combinations are incompatible. The rows of the diagram are referred to as rings, which can be timed independently, so long as both rings cross the barrier at the same time. Note that the separations of the phases between the barriers (e.g. the separation between phases 1 and 2), are based on the interval times assigned to each phase, and have been drawn

arbitrarily in this diagram. Careful observation reveals that there are only eight unique phase combinations that are compatible.

Normally, the cycle is executed in an end-to-end fashion with every phase receiving some interval time in sequence. The only deviation would occur if vehicle detectors were used to skip phases when no vehicles were present. In the proposed control method, the phases have no particular order, and are active based on the queue sizes alone. This does not conflict with the simulation environment, considering we have the benefit of perfect data; however, modifications to the general scheme may be required when applying it to real-world systems where factors such as imperfect vehicle detection, hybrid traffic flows, and side friction must be considered.

Data Constraints

For our purposes, we assume that certain vehicle information is always available, which is a futuristic concept underlying the development of IntelliDrive (formerly Vehicle Infrastructure Integration (VII)). Assuming that every vehicle in the network is running an in-vehicle information system (IVIS) that is capable of communicating with the signal controller in some fashion, we are able to obtain vital traffic information from each vehicle. At the most basic level, we assume knowledge of the vehicle's position in the network. An IVIS-equipped vehicle with a GPS (Global Positioning System) module could easily provide this information in near real-time.

The position of the vehicles is the primary piece of information that we use for this control algorithm, along with whether it is a truck, or not a truck. Other information may include the vehicle's speed, its intended route, or other characteristics. Vehicle speed data is also used to monitor vehicle stops for control method comparisons. The information currently gathered by this system is not complicated, and can be provided by currently available detection systems and GPS tracking systems.

Instead of counting vehicles with a complicated set of detectors, we simply ask the vehicles for their position, and build the queues based on this information. While this simplifies the physical setup of the intersection, the simulation is slowed by having to request information from each vehicle in the network at every step of the simulation. However, this is not entirely unrealistic considering that all of the vehicles near an intersection can already communicate their telemetry data to the signal controller via some wireless communication method. This would enable a process by which all of that information could be gathered to run the control algorithm. It is, of course, entirely possible to use detectors to perform the queue counting function without loss of performance, save the cost of the increased intersection complexity (and the introduction of detection errors). Using the position information to build the queues is thought to be a more generalized solution, because the method can be applied to any intersection without the need for a complicated detector setup.

Evaluation Environment

To test and compare control methods, we have used the VISSIM traffic simulation environment—a microscopic multi-modal traffic simulator that gives the user control over all aspects of the network, such as vehicle type, driver behavior, intersection control, and statistical data collection. The VISSIM simulator allows for many types of signal controllers to be used. Additionally, a built-in fixed-time controller, the VAP (Vehicle Actuated Programming) controller, and ASC/3 SIL (Software-In-the-Loop) controller are implemented in the simulation.

To fully understand the performance of the proposed control technique, it would be helpful to compare it to a current control method. In the results section, we study performance compared to a fixed-time controller, the default controller in VISSIM. Under fixed time control, the active phases change in the same predetermined sequence as in the ring diagram of the Ring Diagram figure, with each phase having fixed interval lengths. This is hardly a fair comparison due to the static nature (even though the times are optimized for the expected volumes) of the fixed-time controller versus the proposed controller, since the latter relies on gathering data pertaining to approaching vehicles and making subsequent informed signal-control decisions. We therefore need to compare this method against another controller that utilizes vehicle information. As such, we chose to use the ASC/3 traffic controller from Econolite. This is an industry standard controller that has many advanced features, the least of which is a vehicle-detection capability.

The control method proposed relies on the algorithm discussed in the section on the Scheduling Algorithm below, the calculations for this are carried out in Matlab. VISSIM provides a COM (Component Object Model) interface in order to give control of the simulator to external applications. Using this COM interface from Matlab, we are able to load the traffic network, set simulation parameters, execute simulations, and collect data. The control routine single-steps through the traffic simulation while controlling the signal group with the custom-designed control logic. A schematic representation of the control interface flow is shown in Figure 3. It is important to note that we have no control over the actual signal controllers; these only interface directly with the simulator.

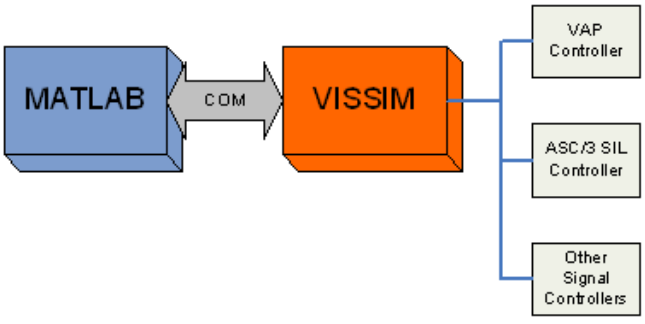

Figure 3. Diagram. Schematic representation of the control interface flow.

Due to nature of the simulator, the COM interface allows for the reading of a signal's state, but does not allow for the changing of the traffic-signal states directly. In order to control the many phases of the intersection, a simple VAP code controller monitors vehicle detectors in the network that are associated with each phase. The VAP controller changes the state of the associated phase whenever the detector is activated. In the simulator, the detectors are disabled to prevent vehicles from triggering them. Instead, when the LQF-MWM controller determines that a phase should be changed, a call is issued from the Matlab environment which, in turn, signals the VAP controller to change the state of the corresponding signal group.

While the ASC/3 makes control decisions every tenth of a second (the resolution of the simulator), the LQF-MWM algorithm only makes control decisions every second. During each control step, the algorithm loops through a list of all of the vehicles in the network and requests information, including position and speed, in order to build the vehicle queues and to determine if any vehicles have stopped. There does not seem to be an appreciable performance difference between the operation of the ASC/3 and the Matlab controllers with respect to the frequency of control decisions. In any case, the less frequent LQF-MWM method is of no advantage to the algorithm.

Chapter 3 – Signal-Scheduling Algorithm

We first consider the phase connection diagram (origin to destination) shown in Figure 4. This diagram indicates which phases are used to move a vehicle through the intersection from any input to any output. (Note: a vehicle cannot leave on the same link from which it arrived.) This data is given to the signal-scheduling algorithm, along with phase compatibility information, in order for it to evaluate the size and weight of each queue. These weights reflect the service urgency of each queue.

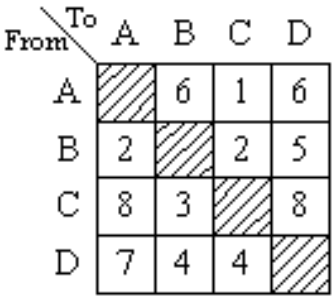

Figure 4. Diagram. Phase connection diagram for the intersection examined.

In the stability discussion below, for the purpose of clarity, the intersection is referred to as a node at which the links (approaches) are connected. This is intended to generalize the proof and to not introduce confusion between physical lanes of the intersection and the overall input-output characteristics of the intersection as a whole. Other terms used in the algorithm's description include traffic flow rate and link capacity. The flow rate is a value that describes how much traffic is flowing on a particular link relative to the overall capacity of the link. The capacity of the link is defined to be the maximum number of vehicles that could possibly traverse a link within a certain amount of time. These quantities are usually described in terms of vehicles per hour.

Longest Queue First Maximal Weight Matching (LQF-MWM) Signal Arbitration

We next describe the proposed signal arbitration algorithm. First, define the traffic load matrix as a doubly sub-stochastic matrix, $\Lambda = \|\lambda_{ij}\|$ with admissible arrival rates, such that

$$\sum_{l=1}^{N} \lambda_{il} < c, \sum_{l=1}^{N} \lambda_{lj} < c, \qquad (1)$$

Figure 5. Equation (1).

where λ_{ij} denotes the average rate of vehicles moving through the intersection from input link *i* destined for output link *j* (*i* is not equal to *j*), *c* the physical capacity of the links, and *N* the number of links that are connected at the intersection node (*N* is 4 in our case). The first part of the equation states that no link has more than its capacity in traffic traversing it. The second part guarantees that none of the destination links will be overloaded. Let $Q(t) = [Q_{11}(t),...,Q_{1N}(t),...,Q_{NN}(t)]^T$ be the queue occupancy vector in which each component represents the number of vehicles currently queued at time *t*. For lanes that are associated with two destinations (e.g. lane 6), we assume an equal queue size distribution between the flows destined to each of the two output links in order to simplify the proof. However, any different distribution would support the proof. Queues are served in accordance with the policy dictated by the signal-control algorithm. Due to the nature of the traffic flow, all $Q_{ii}(t) = 0 \quad \forall i$ (it is assumed that there is no loopback traffic). The signal-control algorithm selects a set of compatible matches between a set of input and output links. The set of matchings is represented by a *matching matrix,* $\|S_{ij}(t)\|, 1 \le i \le N, 1 \le j \le N$, whose binary elements $S_{ij}(t) = 1$ *iff* input link *i* is selected by the control algorithm to connect to output link *j,* otherwise $S_{ij}(t) = 0$.

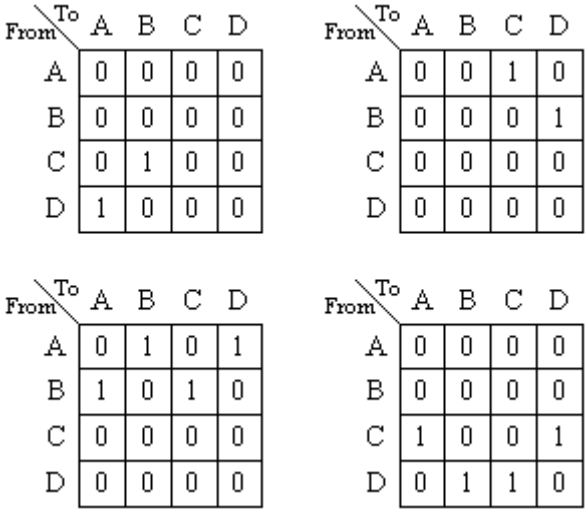

Figure 6. Diagram. Intersection configuration sets considered by the algorithm.

There are four intersection matching matrices considered by the algorithm, as shown in Figure 6. By comparing Figure 6 to Figure 2, and referencing back to the ring diagram of Figure 3, one can see that all possible combinations of the dual-ring phase scheme shown in Figure 3 are "covered" by these matching matrices. In addition, these matching matrices cover all valid link permutation matrices; this is a necessary condition for the validity of the algorithm. Letting the *weight* of a matching be denoted by $W(t) = \langle Q(t), S_{ij}(t) \rangle$, it is noted that given the four configurations of the intersection (i.e. matching matrices) described in Figure 6, there are four corresponding weights, which we label $W_m(t) \quad m = 1,2,3,4$. We further define κ_n $(n = 1,2,3,4)$ as the *sum* of weights corresponding to every combination of three weights $(W_m(t))$. The indices of such combination of weights are $\{1,2,3\},\{1,2,4\},\{1,3,4\},\{2,3,4\}$. The algorithm selects the matching matrix that

has the highest value within the set of weights corresponding to the largest element in κ_n. Note: the algorithm requires the calculation of the weights and κ_n to take place prior to each configuration of the intersection. However, the computational complexity involved in such arithmetic is rather low.

Inherently, the algorithm tends to select lanes with larger queues. However, it is not necessarily the case that the lane corresponding to the largest queue will be selected. This largest queue will be served if, and only if, it is a member of the maximal combination of queues. By choosing the maximal combination of queues, the control method is able to move the greatest number of vehicles through the intersection over time. We further note that an increased measure of priority can be applied to a particular vehicle in the queue by simply giving it an increased individual value. That is, when the queues are being considered by the algorithm, the value of each vehicle can be independently chosen beforehand (based upon vehicle type, for example). By artificially inflating the values of the queues in this manner, we force the algorithm to service queues with high-priority vehicles. Choosing values for the different vehicle classes then becomes a matter of balancing between the maximum queue sizes and the importance of the vehicles considered.

Stability of the Algorithm

In this section, we provide a comprehensive stability proof for the proposed algorithm. At its core, stability implies that the expected value of the queue sizes are all bounded. Another way of expressing this notion is to say that given the proposed signal-control algorithm, the average queue sizes are always strictly bounded. We define $P_k \subset \mathrm{R}^{N\times N}$ as the set of (*N!*) permutation matrices of an *NxN* matrix, i.e. matrices with only a single 1 in each row and in each column. According to Birkhoff's theorem [3], the following inequality holds:

$$\Lambda < \sum_j \alpha_j P_j, \qquad (2)$$

Figure 7. Equation (2).

Where $\sum_j \alpha_j = c.$ The above equation states that any doubly sub-stochastic matrix can be decomposed into a convex sum of permutation matrices.

Let $D(t) = \left[D_{11}(t),\ldots,D_{1N}(t),\ldots,D_{NN}(t)\right]^T$ be a vector denoting the departure process, for which the element $D_{ij}(t)$ represents the number of vehicles departed from link i for *j* link during time slot t. Hence, the evolution of the queue occupancy can be expressed as

$$Q(t+1) = Q(t) + A(t) - D(t), \qquad (3)$$

Figure 8. Equation (3).

where $A(t)$ is the number of vehicles arriving to the queue at time *t*. The intersection under study will be modeled by discrete time queues that, in turn, will be analyzed using Discrete Time Markov Chain (DTMC) models.

Definition: *The weight produced by the LQF-MWM algorithm at time t is given by*

$$W'(t) = \langle Q(t), S'_{ij}(t) \rangle = \sum_{i,j} Q_{ij}(t) S'_{ij}(t), \quad (4)$$

Figure 9. Equation (4).

where $S'_{ij}(t)$ *denotes the matching configurations established by the algorithm at time t. We next provide stability-related definitions that will aid in establishing the stability properties of the algorithm.*

Theorem *(Variation of Foster's Criterion [4]): Given a system of queues whose evolution is described by a DTMC with state vector* $Q(t) \in \mathrm{N}^M$ *, if there exist* $\varepsilon \in \mathrm{R}^+$ *and* $B \in \mathrm{R}^+$ *such that given the function* $L(Q(t)) = Q(t)Q^T(t)$, $||Q(t)|| > B$ *, the following holds:*

$$E[Q(t+1)Q^T(t+1) - Q(t)Q^T(t) \mid Q(t)] < -\epsilon ||Q(t)|| \quad (5)$$

Figure 10. Equation (5).

then the system of queues is <u>*strongly stable*</u>*.*

The LQF-MWM signal-control algorithm determines the configuration of the signals in the intersection once every k time slot units, which defines the switching interval. The latter loosely refers to the number of vehicles that can arrive or depart and would typically be on the order of a few seconds. Next we present the core theorem of this project.

Theorem: *An intersection running the LQF-MWM signal control algorithm with aggregate traffic load destined to any output link that is less than C/3 is stable for any finite switching interval.*

<u>Proof</u>: Since at most k vehicles may arrive during k time slots, the following inequality holds:

$$Q_{ij}(t+k-1) - Q_{ij}(t) \leq k, \qquad (6)$$

Figure 11. Equation (6).

from which we can write

$$Q_{ij}(t+k-1) - Q_{ij}(t) \leq \sum_{-} A_{ij}(t+m) - kS_{ij}(t), \quad (7)$$

Figure 12. Equation (7).

for $Q_{ij}(t) \geq \eta k$. The term $kS_{ij}(t)$ expresses the k consecutive vehicle trips that may occur during a switching interval. Next, we construct a discrete-time quadratic Lyapunov function [5], $L(t)$, defined as $L(t) = \langle Q_t, Q_t \rangle = \sum_{i,j} Q_{ij}^2(t)$. To prove the algorithm yields a stable queueing system, we would like to show that beyond a given threshold of maximum weight there is a negative drift in the state (queue occupancies) of the system. As an expression of a k time slot lag, we can write

$$L(t+k-1) - L(t) = \qquad (8)$$
$$\sum_{ij} \left(Q_{ij}(t+k-1) - Q_{ij}(t)\right)\left(Q_{ij}(t+k-1) + Q_{ij}(t)\right).$$

Figure 13. Equation (8).

For the case of $Q_{ij}(t) \geq k,$ we deduct the following

$$\begin{aligned} & E\left[L(t+k-1) - L(t) | Q(t)\right] \leq \qquad (9) \\ & \leq \sum_{ij} \left(\sum_{m=0}^{k-1} A_{ij}(t+m) - kS_{ij}(t) \right) E\left[2Q_{ij}(t) + k\right] \\ & \leq \sum_{ij} 2E\left[Q_{ij}(t)\right]\left(k\lambda_{ij} - kS_{ij}(t)\right) + \sum_{ij} k^2 \\ & \leq 2k\left[\langle \Lambda, Q_t \rangle - \langle S, Q_t \rangle\right] + k^2 N^2. \end{aligned}$$

Figure 14. Equation (9).

Since we know that

$$\begin{aligned} \langle \Lambda, Q_t \rangle &= \left\langle \sum_j \alpha_j P_j, Q_t \right\rangle = \sum_j \alpha_j \langle P_j, Q_t \rangle \quad (10) \\ &< \sum_j \alpha_j \max_k \langle P_k, Q_t \rangle . \end{aligned}$$

Figure 15. Equation (10).

Given that $\sum_j \alpha_j = 1$ (after normalizing the load matrix), we obtain

$$\left\langle \Lambda, Q_t \right\rangle < \max_k \left\langle P_k, Q_t \right\rangle = \left\langle S^*, Q_t \right\rangle = W^*(t), \quad (11)$$

Figure 16. Equation (11).

which would conclude the proof if all permutation matrices were applicable to the intersection. To evaluate the impact of the partial connectivity that may be applied to the intersection, we note that any permutation matrix can be majorized (or covered) by, at most, three of the allowable intersection configurations. In other words, there exist l, m, and n such that

$$\max_k \langle P_k, Q_t \rangle < \langle R_l, Q_t \rangle + \langle R_m, Q_t \rangle + \langle R_n, Q_t \rangle \quad (12)$$

Figure 17. Equation (12).

for some $l \neq m \neq n \subset \Psi,$ where Ψ is the set of allowable intersection configurations. Because

$$\langle R_l, Q_t \rangle + \langle R_m, Q_t \rangle + \langle R_n, Q_t \rangle < 3 \max_j \langle R_j, Q_t \rangle, \quad (13)$$

Figure 18. Equation (13).

we conclude that $\left\langle \frac{\Lambda}{3}, Q_t \right\rangle < \max_j \left\langle R_j, Q_t \right\rangle$.

This result states that if the *average* aggregate traffic heading to any given output link from all associated input links does not exceed $c/3$ (i.e. a third of the maximal physical capacity of the link), the algorithm will always yield a stable system. Instantaneously exceeding the capacity is acceptable, so long as the average rate is bounded by $c/3$. This is irrespective of the distribution of traffic across the different input links. For the case of quality of service provisioning, the situation does not change, so long as the incoming and outgoing traffic loads remain admissible over time.

Numeric Illustration of the Algorithm

To better illustrate the LQF-MWM control algorithm, a simple example is provided reflecting on the traffic scenario depicted in Figure 19. The weights assigned to the trucks and cars are 20 and 1, respectively, expressing the higher priority given to trucks. A list of pairs and their corresponding weights is shown in Fig. 19. These weights express the summation of on-detector vehicle's weight. From the list, it can be seen that the phase pair 1 and 6 has the highest weight; therefore, phases 6 and 1 will turn green. Compared with the control logic in the ASC/3 controller, which serves the phase pairs sequentially according to the dual-ring phase scheme, the

LQF-MWM yields better service provisioning and the vehicle delay is significantly lower than the one achieved by the ASC/3 control logic.

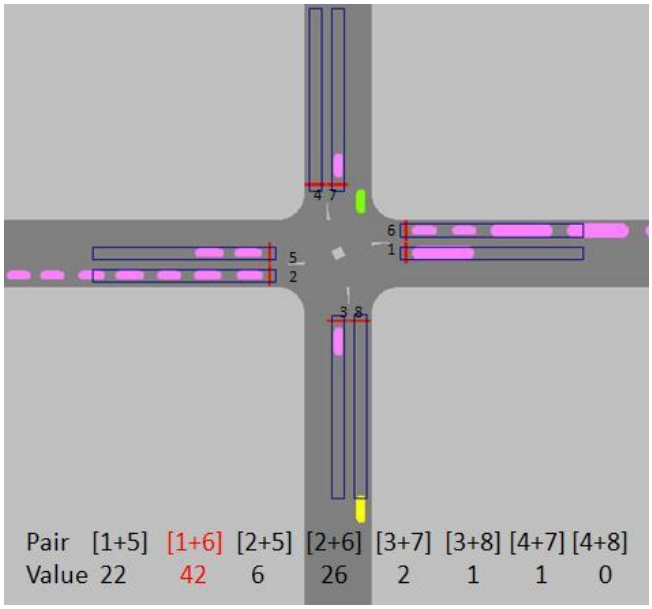

Figure 19. Diagram. Illustration of the LQF-MWM algorithm.

Chapter 4 — Simulation Results

Using the VISSIM simulation environment, the intersection described is investigated under various traffic conditions for multiple control schemes. The primary variable considered is the average traffic load. A value of 1800 vehicles per hour (or one vehicle every two seconds) is taken to be the maximum traffic load for each of the four approaches to the intersection. Incoming traffic load is varied from near zero up to half of the maximum load (c/2). For each data point, an average is taken over three separate simulation runs in order to obtain sufficient statistics. Also, each run simulates 40 minutes of traffic flow. All vehicles within 100 meters of the signal are counted as being in the queue for that signal (as if it were a detector). In addition, both the ASC/3 controller and the Matlab controller use the same minimum and maximum green times. However, the maximum green time is (at most) doubled for the Matlab controller when extensions for high-priority vehicles are used.

When this work was begun, the control schemes were evaluated using only cars in the network. This provided for some uniformity with respect to the behaviors of the vehicles in the network. However, a goal of this effort was to provide some increased measure of service to trucks. Therefore, truck traffic is added to the network in order to compare the per-class quality of service between cars and trucks in the network. The weighting scheme used is such that a queue with a single truck will always outweigh another queue filled completely with cars. This was done only to emphasize the differences in service between the vehicle classes. The exact values of the priorities depend on the properties of the intersection, and are chosen by the operator at design time.

Two types of traffic flows are studied: equal route distribution, in which there is an equal probability for all vehicles to turn left, turn right, or go straight through the intersection; and unequal route distribution, in which there is a more realistic distribution of vehicle destinations. For the latter, the distribution used is 70 percent straight, 20 percent right, and 10 percent left. These values are arbitrary, and are only used to demonstrate the operation of the method. Note: any values may be used, so long as they do not violate the maximal load conditions set forth in the section on the Scheduling Algorithm above. Since the distribution used is the same for all approaches, the average outbound load never exceeds the maximum physical capacity. While this distribution is arbitrarily chosen, it generally represents a more pragmatic traffic pattern. As proven above, any traffic distribution that does not violate the admissibility criteria results in stable operation.

Considered next is the truck traffic load allocated to each approach. Initially, five percent of the traffic is selected as trucks, and this load is appended uniformly to all approaches (and uses the same destination distribution as the cars). This setup results in rather uninteresting behavior because all approaches end up having the same long-term average priority. That is, the resulting prioritized traffic flow experiences the same delay characteristics as the non-prioritized traffic flow.

In order to avoid this, and to more clearly identify the impact of prioritization, an imbalance is enforced such that the approaches receive 0, 5, 10, and 15 percent truck traffic clockwise from the North approach. This means that the North-South directions have 5 percent truck traffic, and the East-West directions have 10 percent truck traffic.

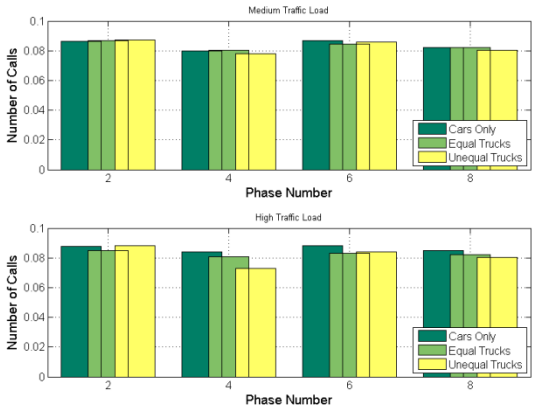

Figure 20. Bar Graphs. Phase selection counts for cars only, homogeneous truck distribution and heterogeneous truck distribution. Heterogeneous truck distribution leads to a larger difference between phase pairs [2+6] and [4+8].

Figure 20 provides a summary of the phase-count percentages, enabling an effective comparison of the different traffic composition scenarios. We observe medium and high traffic loads between three variations of truck distribution: no trucks present, equal truck distribution, and unequal truck distribution. Note: the phase call percentages are fairly well matched between the first two cases. A closer look at Figure 20 reveals that the disparity between phases 2 and 6 (eastbound and westbound) and phases 4 and 8 (southbound and northbound) has become more significant: 2 and 6 have increased while 4 and 8 have decreased, as would be expected.

In addition, differences between the average vehicle delays are also notable. Intuitively, the addition of trucks to the traffic flow should increase the average delay. Trucks are slower to change speed and, therefore, slow the vehicles behind them. Delay characteristics for three cases are shown in Figure 21. Figure 21 indicates that the addition of trucks does increase the average delay. Also, when the approaches are unequally loaded with trucks, the result is another increase in delay. Certainly the most challenging aspect of any scheduling scheme would be to minimize the delay for the worst-case scenario (unequal truck distribution), which we address next.

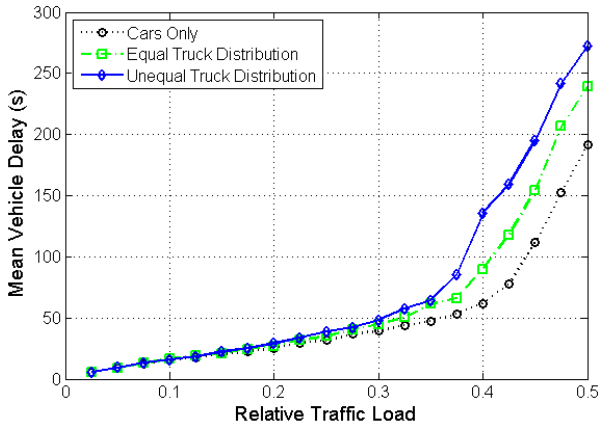

Figure 21. Line Graph. Vehicle delay comparison between three different traffic compositions.

Equal Route Distribution

The first case under consideration is the intersection with equal traffic-route decisions. This means that each vehicle tends to make straight, left turn, and right turn decisions with equal probability. Such a setup gives the most basic form of traffic controller, the fixed-time controller, a best-case scenario for traffic routing. Therefore, we compare our method, along with the ASC/3, to the fixed-time controller. Note: the fixed-time controller has been optimized for the traffic load point (i.e. 0.33 relative traffic load). Two variants of the LQF algorithm are compared to both the fixed-time controller and the ASC/3. One uses no priority, and the weight matrices are based on the queue sizes alone. The other utilizes priority, where the trucks are counted with a higher weight than the cars. This controller also has extensions enabled, whereby the phase interval is extended when there is a high-priority vehicle still in the queue when the interval first comes to an end. The number of extensions is limited to, at most, double the overall maximum green time for that phase interval.

The results of the delay analysis for this intersection configuration are shown in Figures 22 and 23. Figure 22 depicts a comparison of the mean car delays for the four control variants considered. All controllers exhibit similar behavior, and there is almost no difference between them through a wide range of traffic volumes. An exception, of course, is the fixed-time controller, which stands out as a poor performer at low volumes due to its ignorance of the presence of vehicles waiting to be served and its strictly cyclic behavior.

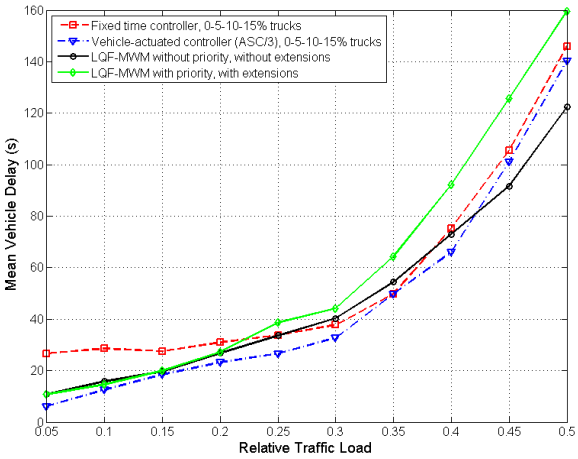

Figure 22. Line Graph. Average vehicle delay for equal route distribution with traffic comprised of cars only.

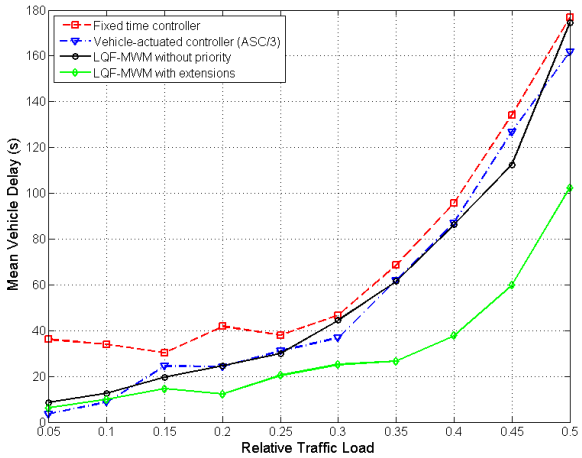

Figure 23. Line Graph. Average vehicle delay for equal route distribution with traffic comprised of trucks only.

Turning now to Figure 23, we observe a larger variation between the different control methods. This figure provides only the average delay of the trucks in the network. In general, the ASC/3 performs marginally better than the other controllers at very low traffic volumes. Again, the fixed-time controller is a poor performer at lower volumes; however, its performance closely follows that of the ASC/3 and the non-prioritized LQF-MWM at medium and high loads for both cars and trucks.

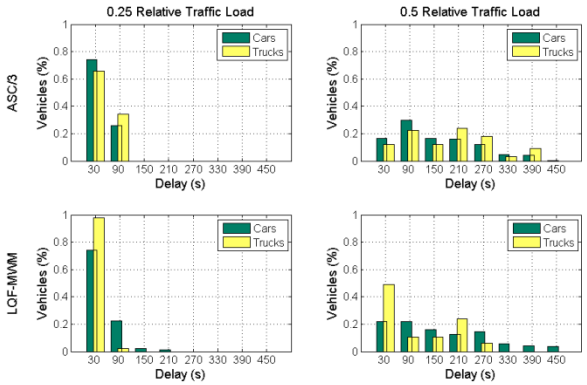

Figure 24. Bar Graphs. Average vehicle delay histogram for equal route distribution at medium and high traffic loads, for the ASC/3 and LQF controllers.

The difference between the algorithms becomes much more apparent when examining the vehicle delay histogram. The latter for equal route distribution is shown in Fig. 24. This shows a comparison between the ASC/3 (op top) and the LQF-MWM controller with priority and extensions (on the bottom). There are two different load points illustrated: the graphs on the left are for (0.25 relative load), and the graphs on the right are for (0.5 relative load). For the lighter loading condition, it can be seen that the ASC/3 delays both classes of vehicles roughly equally. On the other hand, with the prioritized traffic flow of the LQF controller, we see that nearly all truck traffic is delayed less than a minute (the first bar group). The effect of priority on the truck delay is also clear from the histograms for c/2. At this load point, the ASC/3 again delays the traffic fairly evenly. If anything, the cars have a slight edge, given their more-aggressive acceleration profile. With the LQF controller, however, advantage is clearly given to the trucks, at the expense of delaying some cars significantly.

The last performance study pertains to vehicle stops. Recall that a vehicle is considered stopped if it ever comes to rest during its approach or passage through the intersection. Figure 25 shows the results of a comparison between the ASC/3 and the LQF algorithm for equal route distribution. In general, the percentage of vehicles stopped grows as the traffic load increases. As expected, the ASC/3 stops both classes of vehicles almost equally. However, the LQF algorithm, using priority and interval extensions, is able to maintain substantially fewer stops for the trucks across all traffic loads. In fact, the number of stops for both cars and trucks is decreased when using the LQF algorithm. This is due to cars riding along with the trucks through the intersection. At very low traffic volumes, however, the results may not be entirely statistically sound given the limited number of trucks that actually enter the network.

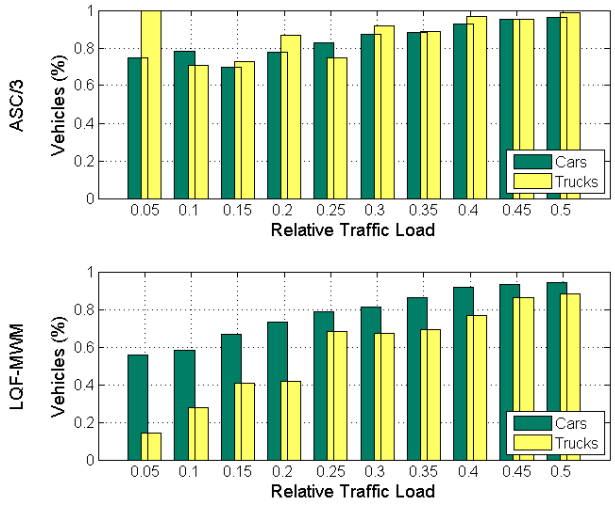

Figure 25. Bar Graphs. Vehicle stops for equal route distribution comparing the ASC/3 controller with LQF-MWM (using priority scheduling and extensions).

Non-uniform Route Distribution

The case in which the traffic routing is unequally distributed is far more realistic. Certainly, the conditions of this isolated intersection are far from reality, but the potential of the control framework may still be evaluated through these results. First, we examine the average vehicle delay for different control methods for the two classes of vehicles under consideration. We compare three variants of our approach to the ASC/3. The first uses no priority, and the weight matrices are based on the queue sizes alone. The second uses priority, but no interval extensions. The third uses both priority and extensions, enabling the latter when a high-priority vehicle is in an active queue when the interval reaches its end. The extensions are made in five-second increments, and the number of extensions is limited to at most double the overall maximum green time for any phase interval. Thus it retains the condition needed for stability, which is a finite expectation on the interval durations.

The car and truck delays are illustrated in Figures 26 and 27, respectively. In Figure 26, we observe that with no priority the performance of the ASC/3 and the LQF algorithm is quite comparable. Moreover, car delays for the two LQF variants with priority are appreciably higher than the ASC/3 delay. This is due to the preference given to servicing trucks. Referring to Figure 27, we observe that truck delay for the LQF algorithms are much lower for all medium and high traffic loads when compared to the ASC/3. The addition of interval extensions does not affect the results substantially, but it does help trucks—and the cars that are around them—to traverse the intersection more quickly than is the case with the ASC/3.

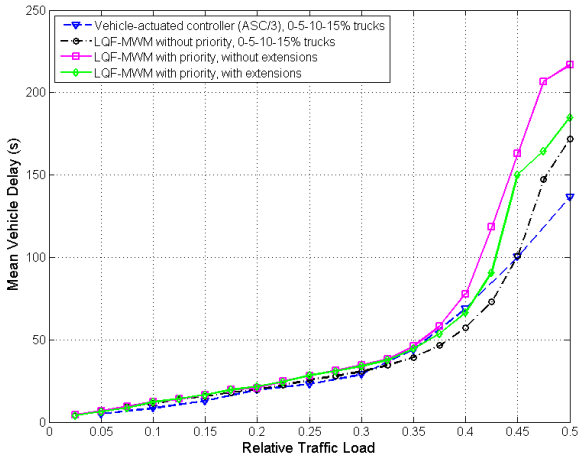

Figure 26. Line Graph. Average vehicle delay for unequal route distribution with traffic comprised of cars only.

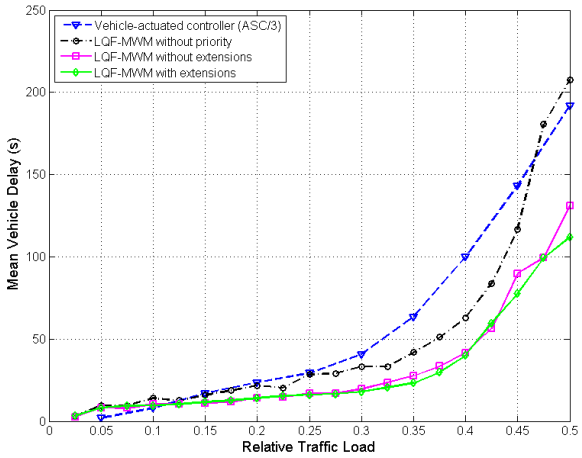

Figure 27. Line Graph. Average vehicle delay for unequal route distribution with traffic comprised of trucks only.

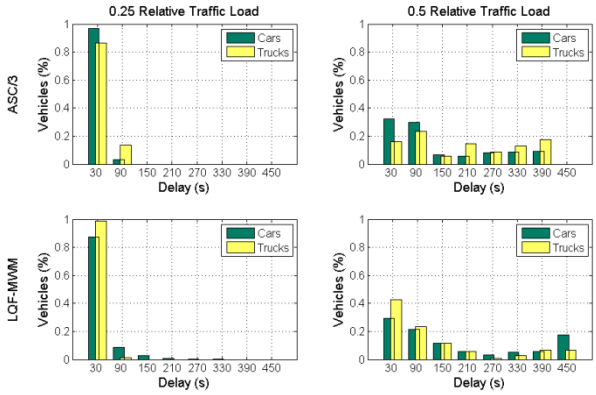

Figure 28. Bar Graphs. Average vehicle delay histogram for various control schemes.

The delay histogram in Figure 28 shows similar results to those found in the equal route distribution case. Again, at low volumes, nearly all truck traffic is ushered through the intersection with minimal delay for the LQF controller, while the ASC/3 delays more than 15 percent of the trucks for more than one minute. At the higher volumes, we see that the LQF controller is able to shift substantial numbers of trucks towards the lower end of the delay spectrum. Referring back to Figure 24, note that the performance is much better for the unequal route distribution case than it is for the equal route distribution case at the lower traffic load. This is explained by the fact that more vehicles are headed straight and right than are turning left. This increases the throughput, allowing more vehicles to traverse the intersection in a shorter amount of time.

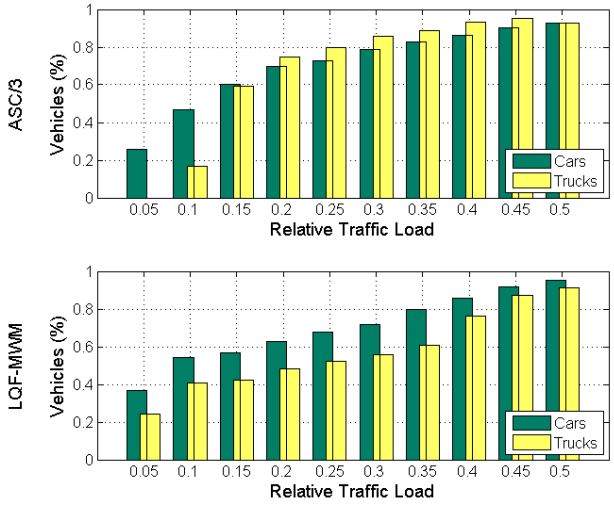

Figure 29. Bar Graphs. Comparison of percentages of trucks that reach a stop for the ASC/3 and LQF-MWM.

Finally, we examine the stops comparison illustrated in Figure 29. The ASC/3 appears to perform much better in this case than in the case of equal route distribution. Again, this is due to its ability to allow the majority of traffic (straight and right) to move through the intersection unimpeded, breaking only briefly to enable the left-turning traffic to move through the intersection. The LQF controller has more stops at the lowest traffic loads, which could be due to a couple of factors. It is most likely due to the small delay that exists between the time the algorithm detects the vehicle in the queue and when it can activate the traffic light. In particular, the ASC/3 reads the detector information ten times per second, while the LQF controller only reads the detectors once every second. This second of added delay decreases the responsiveness of the controller and has a greater affect on the trucks because of their more-moderate acceleration profile. Also, as has been mentioned before, at the lower traffic loads, very few trucks actually pass through the intersection, which results in some statistical inaccuracies. However, the general trends can be identified from the plotted data. Overall, and especially at high traffic loads, the LQF-MWM controller with quality of service provisioning outperforms the ASC/3 by a significant margin.

Chapter 5 — Transition to a field implementable controller

Software-In-the-loop (SIL)

It was necessary to transition from the VISSIM simulation environment with the VAP code controller software interface in order to begin a migration to a NEMA controller with NTCIP. The first step was to test the LQF-MWM algorithm using a virtual ASC/3 controller using software-in-the-loop (SIL) simulation before having to address the system integration issues involving the use of a real ASC/3 controller and NTCIP objects. This initial work involved adding another group of eight detectors to the virtual ASC/3's database. This was achieved by using one group of detectors to detect vehicles and a separate group of detectors to initiate a phase call (see Figure 30). MATLAB was then used to change from an impulse to a constant signal and then to hold on desired phase. The necessary changes in the virtual ASC/3's database included eliminating minimum green time, maximum green time, and the green extension time conflicts. The results from this transition from VAP to SIL were verified to produce the same results as the earlier VAP code controller.

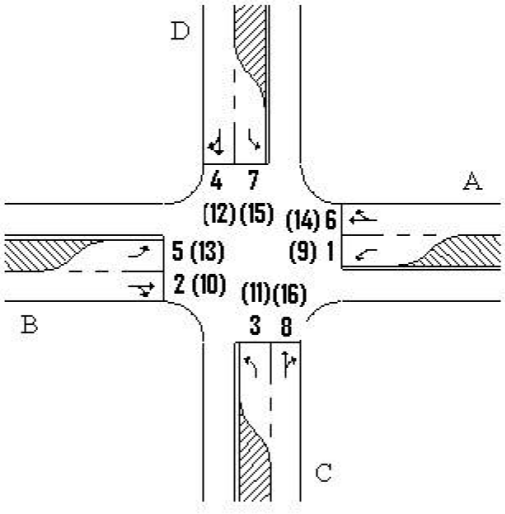

Figure 30. Diagram. Adding two groups of detectors.

Hardware-In-the-loop with detector detecting vehicle status (HIL)

To implement the LQF-WMM algorithm using a real ASC/3 controller, the database in SIL was downloaded to the Econolite ASC/3 controler. The VISSIM traffic environment simulator cannot connect to an ASC/3 controller directly because of real-time synchronization and flash condition issues. An Advanced Traffic Analysis Center Controller Interface Device (ATACid), such as the one shown in Figure 31, was added to address this issue.

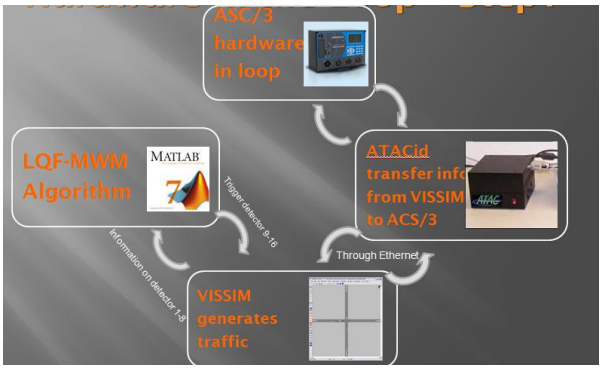

Figure 31. Diagram. Hardware in the loop flow diagram, using ATACid.

The ATACid includes malfunction management logic and the necessary translation between the simulation to keep track of the traffic data in the VISSIM environment. Moreover, it supports SDLC on Port 1 as data bus to update the ASC/3 control box with detector information. To setup the ATACid connection to VISSIM, the configuration setup should be performed prior to the experimental implementation. Upon connecting the modules, a java application displays the real-time phase status.

The operation is identical to the SIL, except we replaced the software ASC/3 controller with the real ASC/3 controller and added an ATACid as a communication interface. The results from this hardware-in-the-loop (HIL) are identical to the SIL results for low and median traffic volume. But for the high traffic volume, the simulation results fluctuated. This was largely due to the vehicle detection method we used. To detect a vehicle in the queue area, we scanned all vehicles in the network and if a vehicle's physical location was in the proximity of a detector area, the vehicle was regarded as in the queue. For example, if we access the vehicle position every 1 second and it takes 60 seconds for a vehicle to pass through the network, then the vehicle position is accessed 60 times during its trajectory. In cases of high traffic volume, the computation time increases exponentially compared to low and median traffic volumes, and becaue in HIL we required real-time simulation, the results became unstable.

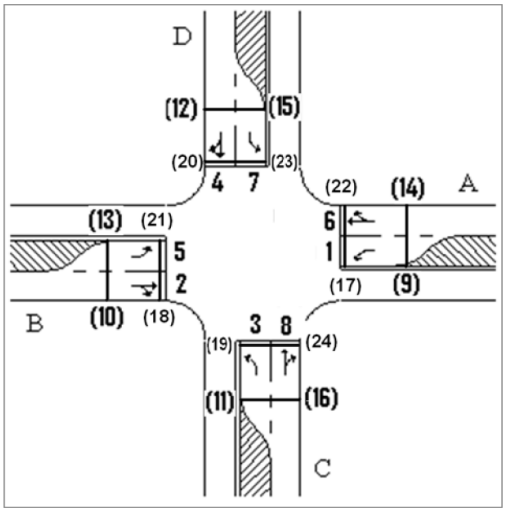

Figure 32. Diagram. Entry and exit detector groups.

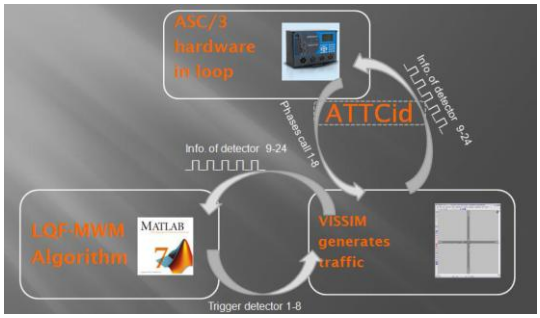

Figure 33. Diagram. Hardware in the loop with entry and exist detector loops.

Hardware-In-the-loop with entry and exit detectors (HIL)

To cope with this challenge of high traffic volume in HIL, we changed our vehicle detection method to counting vehicles by entry and exit detectors, as shown in Figure 32, rather than scanning every vehicle for their positions. Two groups of eight detectors were employed to count the number of vehicles in the queues, with each queue having a pair of detectors. The distance between each pair of detectors is equal to the length of the queue we have designed. The entry detector counts the number of the vehicles that enter the queue, while the exit detector counts the number of vehicles that leave the queue. The queue size is the difference between these two values. This method in Figure 33 dramatically reduced the computation time, compared with the example in HIL: instead of accessing the vehicle position for 60 times, the vehicle information is only accessed 2 times. A third group of eight detectors was used to link to its corresponding phase. These detectors were assigned outside the network to avoid any vehicle passing across. The simulation results are identical to results of SIL, even in the high traffic volume.

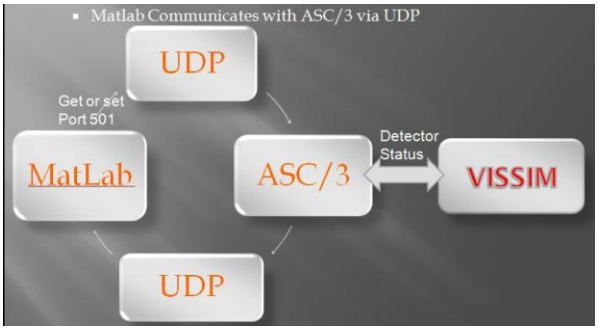

Figure 34. Diagram. Access ASC/3 control box object data by SNMP connection.

Accessing ASC/3 MIB by SNMP

After validation of the algorithm, we wanted to find a way for MATLAB to communicate with the ASC/3 controller box to extract traffic information, as would be necessary in field implementation. The Econolite ASC/3 control box is equipped with NTCIP, which enables access to the objects in ASC/3 traffic database via Simple Network Management Protocol (SNMP). SNMP uses a get-set paradigm to exchange individual pieces of data and an SNMP management station exchanges data by sending each subject-object identifier along with a "get" or "set" request. With the Internet accessibility of ASC/3 control box, we disconnected the link between MATLAB and VISSIM, and connected MATLAB and a physical ASC/3 control box through a network connection User Datagram Protocol (UDP), using port 161, which is specified to SNMP (see Figure 34). After connection was established, we got the detector group's status during every simulation step and set the phase call to activate when a phase change decision was issued.

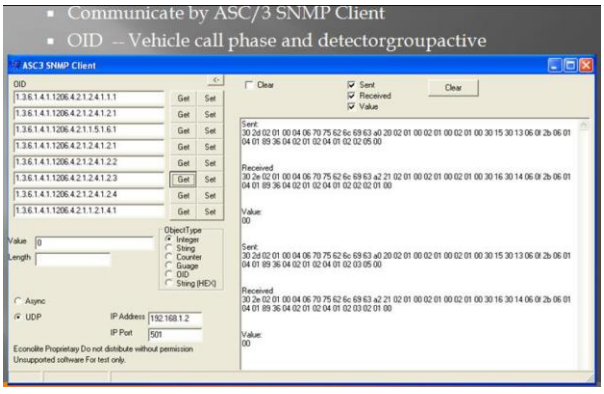

Figure 35. Diagram. Example of ASC/3 to SNMP client access communications.

The traffic objects in ASC/3 are stored within a device called NEMA Management Information Base (MIB), which is a columnar structure and the object identifier of NEMA is 1.3.6.1.4.1.1206. The detector group status object identifier is vehicleDetector-

StatusGroupActive 1.3.6.1.4.1.1206.4.2.1.2.4.1.2 and the phase call is phaseControlGroupVehCall 1.3.6.1.4.1.1206.4.2.1.1.5.1.6 (the ASC/3 OID list has a detail of the OID names). An ASC/3SNMPClient execution file contains the data code, which is required to be sent to the ASC/3 Fig.asc3client. Our Matlab-based controller emulates this ASC/3SNMPClient in order to assess the MIB data object automatically. Before accessing the data, a UDP connection with the Matlab and ASC/3 is established. As is the case in the ATACid IP configuration, the ASC/3 controller box should have the same gateway and mask as the Matlab controller PC.

The response of the SNMP get request is 47 byte packets with the last byte representing the current status of each of the eight detectors. Since the connection between VISSIM and MATLAB has been cut, the only way to detect a vehicle and its vehicle type was by the referring to the time a detector was on. It takes a 13-foot car with a speed of 30 mph 0.43 seconds to cross a 7-foot detector. In SNMP, only one object can be accessed at one time, and the polling frequency for the two groups of detectors was 0.2 seconds. Hence, if a detector was on for 2 to 3 polling cycles, a car was detected. For a 33-foot truck at the same speed, the detector is on for 4 to 6 polling cycles. If a detector is on for longer than 7 polling cycles, there is congestion in the queue. This detection method is easy and efficient, and does not require the deployment of VII.

A potential problem is that the polling frequency is so low that vehicles might pass by the detector before being detected. Because of the redundant overhead in SNMP communication, it normally takes around 0.1 seconds to respond to a request containing approximately 47 bytes of information. Due to the "socket busy" and "transmission error" issues, it is possible for it to take longer than 0.1 seconds to respond. On the agent side of MATLAB, the waiting time for a SNMP response is 0.4 seconds, which means if there is no response from ASC/3 controller in 0.4 seconds, MATLAB will send another "get" request. We might fail to count one vehicle if during these 0.4 seconds a vehicle passes through without being detected. Fortunately, in a service of 500 vehicles, only 2 vehicles went undetected, which does not affect scheduling performance at all.

In this configuration, we were no longer required to set the status of a detector to indirectly activate a phase. Instead, an object phaseControlGroupVehCall enables us to call the desired phases directly. The simulation results are almost the same as the results of the detector counting vehicles.

Accessing ASC/3 MIB by STMP

Concerning the deficit of low polling frequency of SNMP communication, we found a dynamic object configuration in Simple Transportation Management Protocol (STMP) that is able to speed up the polling frequency to 0.1 seconds. STMP, a simplified and more compact version of SNMP, is specially designed to work with dynamic objects that can be regarded as block objects defined by the agent. The data packets can be largely reduced because all the transportation

objects are under the node NEMA 1.3.6.1.4.1.1206 and there is no need to include this overhead when the only communication occurs with the ASC/3 control box. Also, a dynamic object allows the agent to define a sequence of 255 objects. Once this dynamic object is being accessed, the values of the mapping object will be transmitted. This approach improved the polling frequency to 0.1 seconds and significantly reduced the communication bandwidth. As a tradeoff, it increased the complexity of the software within the Matlab agent. A dynamic object containing detector group information has been configured before the program execution. The following is the configuration steps of defining two detector group status objects to dynamic object 3 (Figure 36):

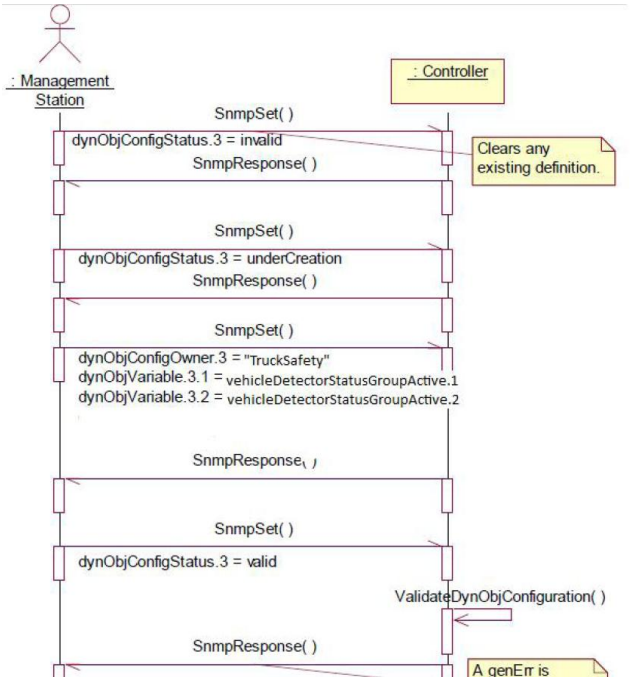

Figure 36. Diagram. STMP Dynamic object setup protocol.

By substituting SNMP to STMP, we can avoid the missing traffic information. The simulation results derived from STMP dynamic object are almost identical to the results of SNMP. We have thus successfully implemented the LQF-MWM scheduling algorithm using a real ASC/3 control box, operated externally from VISSIM and using NTCIP objects.

Signal Control and Prioritization

The truck-safety priority service can be incorporated into a real traffic system. To realize the dynamics and flexibility of the communication between the trucks, a GPS system is embedded in the trucks to generate the real-time traffic data, such as location, speed, vehicle type, and direction. Two components are added to systematize the priority service. First, messages from vehicles are received by the priority request generator (PRG). The PRG processes each message

and checks the necessity for generating a priority service. If a priority request is generated, it will send the priority request service center (PRS) a message to generate a request service, and the traffic control will process the service according to the priority level, as illustrated in Figure 37.

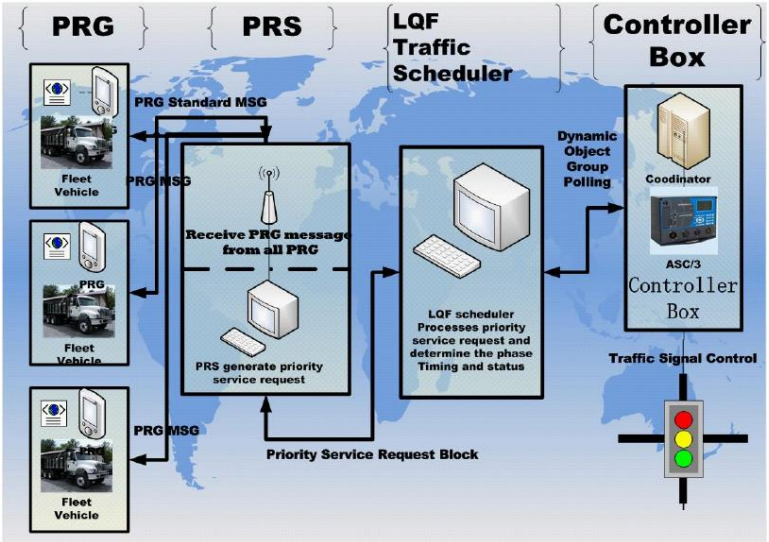

Figure 37. Diagram. Signal control and prioritization.

Chapter 6 — Conclusions

The project successfully demonstrated a novel traffic-scheduling algorithm, implemented using NTCIP, making the concept portable to multiple vendors. The scheme is adaptable with future flexibility in place as more experience is obtained, and does not require any change to existing traffic-control infrastructure. In cases of high traffic volume, the LQF-MWM algorithm yields an average vehicle delay that is half of that observed when a traditional National Electrical Manufactures Association (NEMA) controller operates the signals using standard NEMA sequencing. For example, assuming 900 vehicles/hour/approach in unequal route distribution, the average truck delay for the LQF-MWM algorithm is 110 seconds, while use of the traditional controller logic results in 195 seconds. Because the LWF-MWM algorithm gives priority to trucks, they experience very low delay and stop less frequently at intersections. However, cars experience a slightly higher delay than that observed when using the traditional controller logic. Assuming, once again, 900 vehicles/hour/approach, the average car delay for LQF algorithms is 180 second while it is only 140 seconds for traditional controller logic. In summary, this two-year effort successfully achieved its objective in assessing the feasibility of collaborative heavy vehicle/traffic signal operation using real control-logic hardware. It is concluded that improved heavy-traffic-flow efficiency and safety is attainable with the proposed framework.

Chapter 7 — References

[1] J. D. Little, *A Proof for Queuing Formula: $L = \lambda W$.* Operation Research Vol. 9, 1961.

[2] W. A. Rosenkrantz, "Little's theorem: a stochastic integral approach," *Queuing Syst. Theory Appl.,* vol. 12, no. 3-4, pp. 319-324, 1992.

[3] G. Birkhoff, "Three observations on linear algebra," in *Univ. Nac. Tucum'n. Rev. Ser. A,* vol. 5, pp. 147-151, 1946.

[4] F. G. Foster, "On the stochastic matrices associated with certain queuing processes," in *Ann. Math Statist,* vol. 24, pp. 355-360, 1953.

[5] L. R. A. Bacciotti, *Liapunov Functions and Stability in Control Theory, second ed.* Berlin: Springer, 2005.